game environment or virtual simulation environment to the user. The display device **110** is driven or controlled by the one or more GPUs **106** and optionally the CPU **104**. The GPU **106** processes aspects of graphical output that assists in speeding up rendering of output through the display device **110**.

The ECS device **101** also includes a memory **102** configured to store a game engine **112** (e.g., executed by the CPU **104** or GPU **106**) that communicates with the display device **110** and also with other hardware such as the input device(s) **108** to present a game (e.g., video game) or simulation to a user (not shown in the Figure). The game engine **112** would typically include a physics engine, collision detection, rendering, networking, sound, animation, and the like in order to provide the user with a video game (or simulation) environment. The game engine **112** includes an ECS module **114** that provides various entity component system functionality as described herein. Each of the ECS module **114**, and game engine **112** include computer-executable instructions residing in the memory **102** that are executed by the CPU **104** and optionally with the GPU **106** during operation. The ECS module **114** may be integrated directly within the game engine **112**, or may be implemented as an external piece of software (e.g., a plugin).

In accordance with an embodiment, the ECS module **114**, executing on the ECS device **101**, may be configured to create and manipulate an entity, which includes data, and which is a representation of a game object within a scene of a video game (or simulation). The entity can represent any game object (e.g., any virtual object within a game or simulation) including characters, props, scenery and effects. The entity includes data (e.g., entity data) that describes all aspects, properties and behaviors of the game object which it represents over time. The data includes data describing the visual aspects (texture, color, size, shape, orientation and the like) of the game object; and the data includes data describing the behavior for the game object (e.g., movement of the object and the physics of interaction with other objects in the environment). The behavior of an entity is defined by the processes (e.g., functions) that modifies data of an entity.

In accordance with an embodiment, the entity data includes one or more small groups of data referred to herein as component data. In accordance with an embodiment, during execution (e.g., at runtime during game play) the ECS module **114** creates a component for an entity within a data value array structure (e.g., a 'struct' from within the C# programing language), wherein the elements within the array are laid out in contiguous memory blocks within the memory **102**. A component does not contain a pointer to data in other distant locations within a memory **102**. A component includes data that is associated with a logical grouping of data and behaviors which are used for adding functionality to a single entity. A component can add any type of functionality to an entity, including visual attributes and interaction with other components (e.g., within the same entity or within a different entity). The combination of components within an entity, and the data within the components, contribute to the properties and functionality of the entity in the game world during game play. For example, there can be a camera component which gives an entity the properties of a camera. There can be a light component which gives an entity the properties of a light. For example, a component could define the position, rotation and scale of an entity within a game world. For simplicity of explanation, we will refer to the component that defines the position, rotation and scale of an entity as the transform component since modifying the transform component of an entity would move, rotate or scale the entity (i.e., transform it) within the game world. As another example of a component, a component referred to herein as a rigidbody component could enable physical behavior for an entity by allowing the entity to be affected by gravity within the game world. Still another example of a component could be a component, referred to herein as a collider component, that defines the shape of an entity for the purposes of a physical collision with one or more other entities.

In a typical game or simulation, a plurality of entities have some overlap in the type of components they contain (e.g., two or more entities will have one or more components of the same type). For example, consider a game that includes five entities within a scene and wherein each entity has a transform component (e.g., with the transform data being independent for each entity). In accordance with an embodiment, when two or more entities contain the exact same number and type of components, the entities are referred to herein as an archetype. All entities with the same archetype have the same number and type of components and therefore share similarities with respect to the area which they occupy in memory **102**. However, even though all entities with the same archetype have the same number and type of components, the specific component data for an entity is independent (and usually different) from the other entities. In accordance with an embodiment, the ECS module **114** groups (e.g., places) a plurality of entities of an archetype (e.g., all the entities of the archetype) contiguously together in memory **102** (e.g., as described with respect to FIG. **2**A, **2**B, **3** and with respect to the methods described in FIGS. **4**A, **4**B and **4**C). A location in memory **102** where the plurality of entities of a single archetype are grouped together is referred to herein as a chunk. A chunk is a contiguous block (e.g., a section or area) within memory **102** containing entities sharing the same archetype. In accordance with some embodiments, a single archetype is contained within a single chunk. In accordance with other embodiments, a single archetype can be divided into two or more chunks if a single chunk is not large enough to contain the archetype. In accordance with an embodiment a chunk has a fixed size in memory (e.g., 16 kilobytes or 64 kilobytes)

In accordance with an embodiment, and shown in FIG. **2**A, is a schematic diagram of a data layout for a chunk **200** in memory **102**. Data within a chunk **200** is divided (e.g., by the ECS module **114**) into a plurality of sections, wherein a section contains the data for a single type of component (e.g., a transform component) for all entities in the archetype associated with the chunk **200**. In some embodiments the data within a section is created by the ECS module **114** within a data value structure such as an array. Throughout the description herein, an array which contains all data within a section (e.g., for a component type) is referred to as a component data array. In accordance with an embodiment, and shown in FIG. **2**A, the plurality of different component data arrays within a chunk **200** are placed by the ECS contiguously in memory **102** so that all the component data is laid out linearly and compact (e.g., contiguously) within memory **102**. FIG. **2**A shows an example wherein a chunk **200** contains an archetype that has a plurality of entities (e.g., 5 entities) that all contain three components: a first component (component 'A'), a second component (component ('B'), and a third component (component 'C'). The data for component A is placed by the ECS module **114** in a first data array in a first section **204**A. The data for component B is in a second data array in a second section **204**B. The data for component C is placed by the ECS module **114** in a third